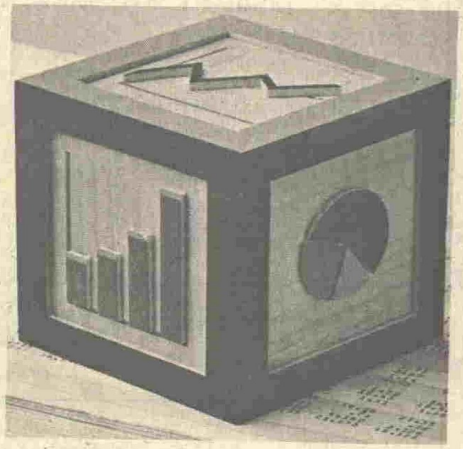
ยืน ภู่วรวรรณ

# สร้างกราฟสวยให้ดีเบสด้วย โปรแกรม FLIPPER

**ผู้ช่วยดีเบสอีกรายหนึ่ง ที่สนับสนุนการสร้างกราฟ ได้หลายรูปแบบ จนต้องทึ่งฝีมือตัวเอง**



ในไมโครฉบับที่แล้วเราได้กล่าวถึงโปรแกรม UI เป็นซอฟต์แวร์สนับสนุนดีเบส สำหรับฉบับนี้จะเสนอซอฟต์แวร์สนับสนุนดีเบสอีกตัวหนึ่งชื่อว่า FLIPPER ที่บริษัทโปรเวอร์กเป็นผู้พัฒนา โดยมีจุดมุ่งหมายในการสนับสนุนการสร้างกราฟให้กับดีเบส อันเป็นสิ่งที่ผู้เขียนโปรแกรมดีเบสหลายคนต้องการ ฟลิปเปอร์จึงเป็นโปรแกรมสนับสนุนดีเบสที่น่าสนใจมากโปรแกรมหนึ่งที่ผู้ใช้ดีเบสควรทดลองใช้

ฟลิปเปอร์ประกอบด้วยดิสค์ขนาด 360 กิโลไบต์จำนวน 4 แผ่น แต่ละแผ่นมีชื่อและข้อมูลต่อไปนี้

- แผ่นที่ 1 ชื่อ Graphics Environment ประกอบด้วยแฟ้มสำคัญได้แก่ README.IST, FLIRPER.EXE และแฟ้ม GRAPHPLS.EXE
- แผ่นที่ 2 ชื่อ FontDisk1 มีแฟ้มที่สำคัญคือ EDITOR.EXE และ FONT2.EXE
- แผ่นที่ 3 ชื่อ FontDisk2 มีแฟ้มที่สำคัญคือ FONT1.EXE
- แผ่นที่ 4 ชื่อ Overlay Manager มีแฟ้มชื่อ FOVERLAY.EXE และ OBJECT40.EXE

ทุกแผ่นที่เก็บข้อมูลจะมีแฟ้ม .EXE ทำหน้าที่จัดแฟ้มอื่น ๆ ในลักษณะลดขนาดพื้นที่ให้เล็กลง เวลาใช้งานจะต้องขยายหรือแยกแฟ้มออกมา โดยการเรียกชื่อแฟ้ม .EXE เช่น สร้างไดเรคทอรีย่อยไว้ในฮาร์ดดิสค์ไดรฟ์ C เพื่อเข้าสู่ไดเรคทอรีย่อยนั้น แล้วใส่ดิสค์แผ่นแรกคือ Graphics Environment ในช่องไดรฟ์ A

```
C:>a: FLIPPER
```

การกระทำในลักษณะนี้จะแยกออกเป็นแฟ้มย่อย ซึ่งแฟ้มที่สำคัญคือ FLIPPER.LIB และตัวอย่างที่ทางบริษัทผู้ผลิตเตรียมไว้ให้อีกหลายแฟ้ม ส่วน GRAPHPLS.EXE ที่อยู่ในแผ่นดิสค์จะเก็บแฟ้ม GRAFPLUS.EXE สำหรับเรียกใช้ในกรณีที่ต้องการพิมพ์รูปกราฟออกทางเครื่องพิมพ์ ฟลิปเปอร์เป็นโปรแกรมประเภทฝังตัวเองลงในหน่วยความจำของเครื่องและเรียกใช้เป็นไดรเวอร์สำหรับเครื่องพิมพ์

ส่วน EDITOR.EXE ประกอบด้วยแฟ้มย่อยในการสร้างตัวอักษรและโปรแกรมยูทิลิตี้อื่น ๆ ส่วน FONT1.EXE และ FONT2.EXE เป็นที่เก็บของรูปแบบตัวอักษร

OBJECT40 และ FOVERLAY.EXE จัดเก็บรวบรวมโปรแกรมและส่วนที่จะใช้สำหรับการลิงก์แบบโอเวอร์เลย์ ขณะเริ่มต้นใช้งานถ้าเขียนโปรแกรมเรียก FLIPPER ไม่ใหญ่โตนักก็ยังไม่จำเป็นต้องใช้โอเวอร์เลย์

## ไลบรารีที่ต้องลิงก์

ความเป็นจริงแล้วหัวใจสำคัญของ FLIPPER อยู่ที่ FLIPPER.LIB ซึ่งเป็นไลบรารีเก็บฟังก์ชันและโมดูลต่าง ๆ ไว้มากมายเกี่ยวกับการสร้างกราฟิกที่เป็นกราฟและรูปภาพ

เราลองพิจารณาโปรแกรมตัวอย่างเพื่อทำความเข้าใจวิธีการทำงานของ FLIPPER โดยเริ่มจากข้อมูลที่เขียนขึ้นด้วยดีเบสดังรูปที่ 1

เมื่อเขียนโปรแกรมเรียกให้ฟลิปเปอร์นำข้อมูล xnumber และ number1 ไปพล็อตเป็นกราฟเส้นตรง ลักษณะของโปรแกรมที่เขียนขึ้นมีคำสั่งของฟลิปเปอร์ประกอบอยู่ด้วย คำสั่งต่าง ๆ เหล่านี้เป็นฟังก์ชันหรือโมดูลที่เก็บไว้ในไลบรารี และจะเรียกใช้เมื่อต้องการลิงก์โปรแกรม ลองดูตัวอย่างโปรแกรมในรูปที่ 2 สมมติเราเก็บข้อมูลโปรแกรมนี้ไว้ในแฟ้มที่ชื่อ EXAM01.PRG

```
. use test
. list stru
Structure for database: B:\TEST.DBF
Number of data records:      5
Date of last update    : 04/29/91
Field  Field Name  Type        Width     Dec
    1  XCHARACTER  Character      15
    2  XDATE       Date            8
    3  XNUMBER     Numeric         8       2
    4  NUMBER1     Numeric         8       2
** Total **                       40

. list
Record#  XCHARACTER     XDATE      XNUMBER   NUMBER1
      1  DBASE          03/01/86      3.00      0.45
      2  LOTUS          03/10/86      5.00      1.20
      3  NORTON         03/25/86      6.50      2.24
      4  WORD           04/10/86      8.30      4.20
      5  HG             04/30/86     12.00      6.10
```

รูปที่ 1 ข้อมูลตัวอย่างที่เขียนขึ้นด้วยดีเบส

```
. set print off* EXAM01.PRG
FLIP_INIT(6000)
USE TEST
INITDATA(2)
SET_TYPE(1,1)
DO WHILE .NOT. EOF()
   @ 23,0 SAY STORE_DATA(XNUMBER,NUMBER1)
   SKIP
ENDDO
PLOT()
inkey(0)
TEXTMODE()
RETURN
```

รูปที่ 2 ตัวอย่างโปรแกรมพล็อตกราฟเส้นตรงด้วยคำสั่งในฟลิปเปอร์

## ใช้คลิปเปอร์คอมพายล์

โปรแกรมนี้ไม่สามารถเรียกด้วยดีเบสโดยตรง ต้องใช้คลิปเปอร์คอมพายล์โปรแกรม โดยมีขั้นตอนดังนี้

```
C:>CLIPPER EXAM01.PRG
```

เมื่อโปรแกรมถามชื่อ .OBJ ให้กดปุ่มรีเทอร์นตลอด ผลการคอมพายล์จะได้ EXAM01.OBJ (ในกรณีที่ไม่พบข้อผิดพลาดระหว่างการคอมพายล์) หากพบข้อผิดพลาดคลิปเปอร์จะรายงานข้อผิดพลาดนั้นออกมา ผลที่ได้นี้เป็นเพียงออบเจ็คต์ ซึ่งไม่สามารถเรียกใช้งานได้ ต้องลิงก์กับโมดูลย่อยที่เก็บไว้ในไลบรารีก่อน การลิงก์ในไลบรารีนี้เราสามารถใช้ PLINK86 ซึ่งเป็นโปรแกรมลิงก์ที่ติดมากับคลิปเปอร์ก็ได้ โดยใช้คำสั่ง

```
C:>PLINK86 FI EXAM01 LIB FLIP88
   LIB CLIPPER, EXTEND
```

การใช้ PLINK86 นี้ทำงานช้าและต้องการพื้นที่ว่างบนฮาร์ดดิสก์ หากเป็นโมดูลขนาดใหญ่คือมี .PRG เรียกไปมาหลายชั้น การลิงก์ก็จะช้ายิ่งขึ้น ดังนั้นนักเขียนโปรแกรมดีเบสส่วนใหญ่จึงใช้ลิงก์ที่ชื่อ LINK บนเอ็มเอสดอสหรือคอมพายเลอร์อื่น เช่น C หรือมาโครแอสเซมบลี LINK ที่ใช้ควรเริ่มจากเวอร์ชันที่ 3.61 หรือดีกว่า คำสั่งลิงก์ของไมโครซอฟต์ใช้งานคล้ายกับของ PLINK86 โดยใช้คำสั่ง

```
C:>LINK/SE:256/NOE EXAM01,,,
   FLIP88+CLIPPER+EXTEND
```

เมื่อลิงก์เรียบร้อยแล้วจะได้โปรแกรม EXAM01.EXE ที่สามารถเรียกทำงานได้อย่างอิสระเมื่อเรียกโปรแกรมนี้

```
C:>EXAM01
```

ผลที่ได้จะแสดงกราฟบนจอภาพดังรูปที่ 3

## รายละเอียดของโปรแกรม

เริ่มคำสั่งแรกด้วย FLIP_INIT (6000) เป็นคำสั่งที่เรียกใช้เพียงครั้งเดียว เพื่อกำหนดขนาดบัฟเฟอร์สำหรับการสร้างกราฟ ขนาดของบัฟเฟอร์นี้ขึ้นอยู่กับจอภาพและการ์ดแสดงผลที่ใช้ หากต้องการเรียกใช้ FLIP_INIT มากกว่าหนึ่งครั้ง เราต้องทำการปลดหน่วยความจำออกเสียก่อนด้วยคำสั่ง FREE _BUF ขนาดของบัฟเฟอร์ขึ้นกับข้อมูลที่เก็บก่อนการพล็อตกราฟ กล่าวคือข้อมูล X, Y ที่จะพล็อตนั่นเอง โดยข้อมูลนี้จะใช้ภาษาซีจำนวน 8 ไบต์ต่อค่า ดังนั้น 6000 ไบต์หาร 8 จะได้ 750 ค่า หรือจุดของการพล็อตจะได้ 375 คู่ของข้อมูล X, Y นั่นเอง

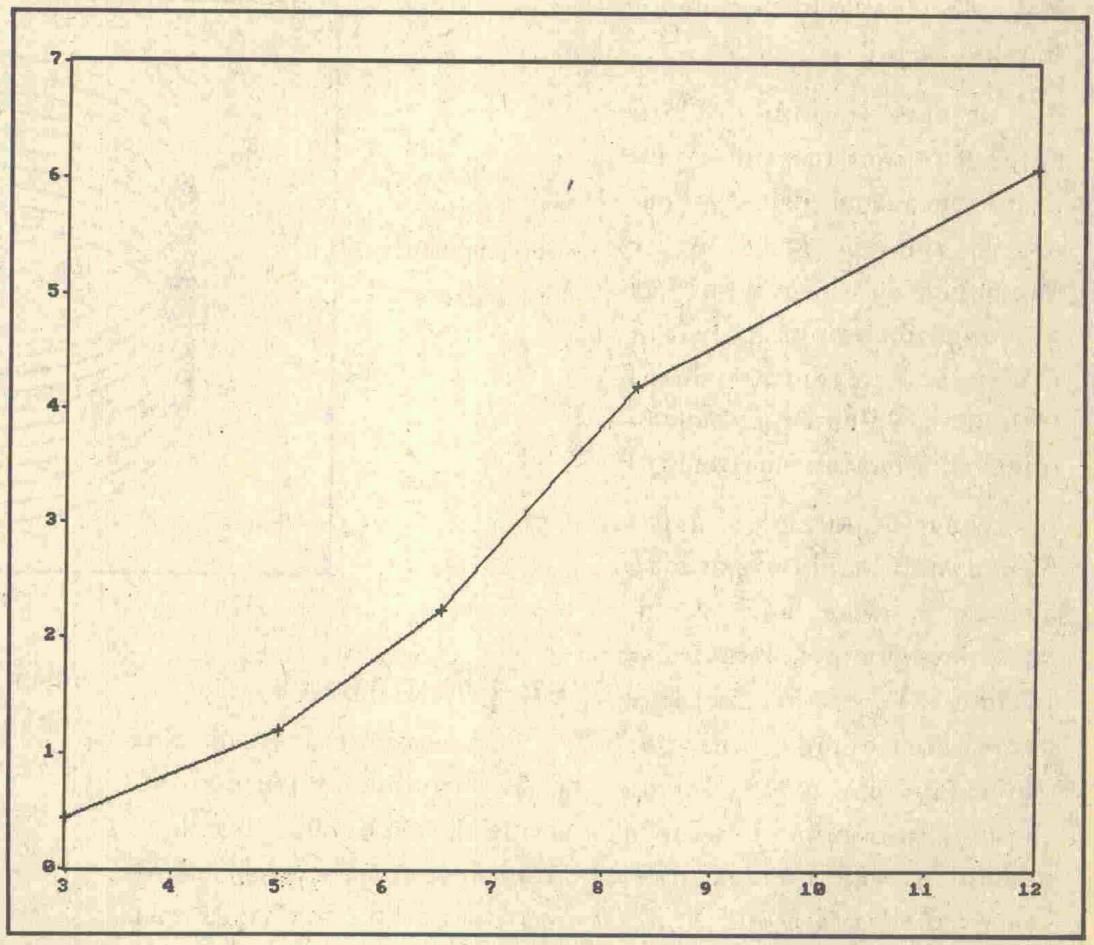
INITDATA(2) บอกจำนวนคอลัมน์ของข้อมูลที่จะเก็บในบัฟเฟอร์ว่ามีขนาดกี่คอลัมน์ ในที่นี้เก็บ 2 ค่า คือ X, Y ค่าที่เก็บได้สูงสุดมี 7 คอลัมน์

SET_TYPE(1,1) เป็นคำสั่งกำหนดรูปร่างของกราฟที่จะให้แสดง เช่น กราฟเส้น พล็อตจุดอย่างเดียว กราฟแท่ง สแต็ก วงกลม ไฮ-โลว์ ค่าของคอลัมน์ Y มี 6 ค่าหรือ 6 ตัวแปร เมื่อเทียบกับ X ในที่นี้พารามิเตอร์แรกบอกว่ามีค่า Y เพียงตัวเดียว พารามิเตอร์ที่สองเป็น 1 บอกว่าเป็นกราฟเส้น

STORE_DATA(X,Y) เป็นการนำข้อมูลจากแฟ้มในที่นี้คือ TEST.DBF มาเก็บไว้ในตัวแปร X, Y โดย X คือค่า XNUMBER และ Y คือ NUMBER1 ค่าของการนำข้อมูลเข้ามาจะต้องเป็นไปตามลักษณะที่มีการกำหนดไว้ก่อนด้วยคำสั่ง INITDATA ในที่นี้ใช้ลูปเพื่ออ่านข้อมูลเข้าบัฟเฟอร์ทั้งหมด

PLOT( ) คำสั่งเขียนกราฟปรากฏออกมาทางจอภาพ โดยรูปกราฟและข้อมูลต้องเป็นไปตามที่กำหนด

INKEY (0) เป็นคำสั่งให้โปร-




รูปที่ 3 รูปกราฟบนจอภาพหลังจากการลิงก์

แกรมแสดงกราฟบนจอภาพ แล้วรอจนกว่าจะมีการรับแป้นพิมพ์อะไรก็ได้จึงค่อยทำงานต่อไป

TEXTMODE( ) เป็นคำสั่งกลับเข้าไปในโหมดแสดงผลตัวอักษรแบบปกติเพื่อทำงานอื่น

เราจะสังเกตได้ว่าโปรแกรมที่เขียนขึ้นนี้เป็นโปรแกรมที่ใช้สำหรับสร้างกราฟทุกรูปแบบ ผู้ใช้สามารถสร้างกราฟจากข้อมูลของดีเบสได้เอง ซึ่งเป็นเรื่องที่มีประโยชน์มาก

```
* EXAM02.PRG
FLIP_INIT(1000)
M_X=100
M_Y=100
M_RADIUS=50
DO WHILE .T.
   CLEAR
   @ 23,70 say 'Esc-exit'
   @ 10,10 SAY 'X:      'GET M_X PICT '999'
   @ 11,10 SAY 'Y:      'GET M_Y PICT '999'
   @ 12,10 SAY 'RADIUS: 'GET M_RADIUS PICT '999'
   READ
   IF LASTKEY()=27
      EXIT
   ENDIF
   FSCREEN()
   FOR i = 1 to 10
      FCIRCLE(M_X+10*i,M_Y,M_RADIUS,i)
   NEXT
   INKEY(0)
   TEXTMODE()
ENDDO
RETURN
```

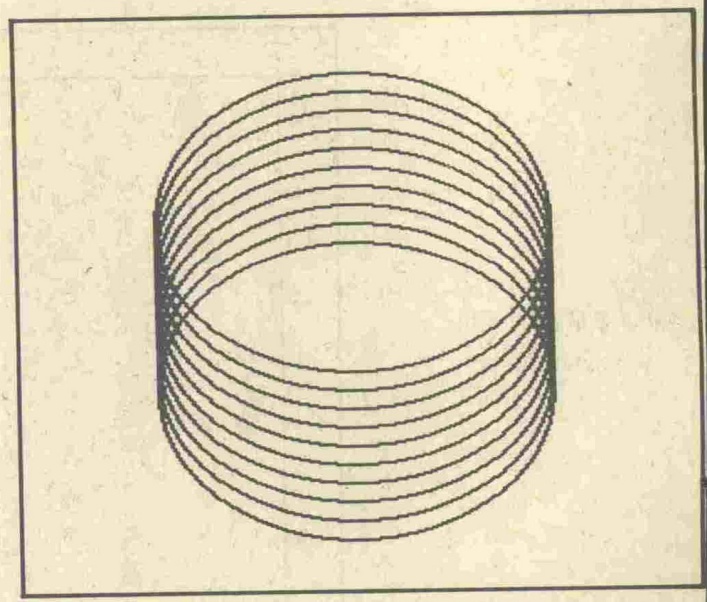
รูปที่ 4 โปรแกรมเขียนวงกลมหลาย ๆ รูป

## อีกสักตัวอย่างหนึ่ง

ตัวอย่างที่สองนี้เป็นตัวอย่างของการเขียนรูปวงกลมหลาย ๆ วงบนจอภาพ รายละเอียดของโปรแกรมที่เขียนได้แสดงไว้ในรูปที่ 4

คราวนี้เรากำหนดบัฟเฟอร์ไว้ใน FLIP_INIT เพียง 1000 หลังจากนั้นกำหนดตัวแปรแล้วเข้าลูปรับค่าจุดศูนย์กลางของวงกลมคือ M_X, M_Y รับค่ารัศมีและสีเก็บไว้ในบัฟเฟอร์ และตรวจสอบว่ามีการกดปุ่ม Esc หรือไม่ ถ้ามีก็จะเลิก หลังจากนั้นจะมีคำสั่ง FSCREEN( ) เป็นคำสั่งบอกให้เปลี่ยนเข้าสู่กราฟิกเพื่อเตรียมทำการพล็อตรูป

คำสั่ง FCIRCLE( ) คือการเขียนรูปวงกลมโดยใช้พารามิเตอร์ถึง 4 ตัวคือ X, Y, radios, และ color ค่า X, Y คือจุดศูนย์กลาง ค่าของรัศมีจะได้จาก 0-154 ส่วนสีจะเป็นไปตามมาตรฐานของไอบีเอ็ม ในที่นี้เราจะเขียนวงกลมจำนวน 10 รูป โดยเลื่อนจุดศูนย์กลางเพิ่มขึ้นทีละ 10 หน่วยไปทางแกน X ผลที่ได้ตรงส่วนบนของจอภาพจะเป็นดังรูปที่ 5



รูปที่ 5
ผลบนจอภาพจากการรันโปรแกรมเขียนวงกลม

## อื่น ๆ อีกมากมาย

ตัวอย่างที่กล่าวทั้งหมดเป็นเพียงสิ่งที่ชี้ให้เห็นขีดความสามารถขั้นต้น ความจริงแล้วเราจะเขียนรูปกราฟและตัวหนังสือของเราเองได้ แม้แต่ตัวหนังสือภาษาไทย เราก็สามารถสร้างรูปแบบตัวหนังสือใหม่ได้ รายละเอียดคำสั่งของฟลิปเปอร์จะอธิบายให้ทราบพร้อมตัวอย่างประกอบเพิ่มเติมในฉบับต่อไป